Sparsity-Inducing Binarized Neural Networks

Peisong Wang,* Xiangyu He,* Gang Li, Tianli Zhao, Jian Cheng†
Institute of Automation, Chinese Academy of Sciences, Beijing, China
University of Chinese Academy of Sciences, Beijing, China
{peisong.wang, xiangyu.he, jcheng}@nlpr.ia.ac.cn

Abstract

Binarization of feature representation is critical for Binarized Neural Networks (BNNs). Currently, $sign$ function is the commonly used method for feature binarization. Although it works well on small datasets, the performance on ImageNet remains unsatisfied. Previous methods mainly focus on minimizing quantization error, improving the training strategies and decomposing each convolution layer into several binary convolution modules. However, whether $sign$ is the only option for binarization has been largely overlooked. In this work, we propose the Sparsity-inducing Binarized Neural Network (Si-BNN), to quantize the activations to be either 0 or $+1$, which introduces sparsity into binary representation. We further introduce trainable thresholds into the backward function of binarization to guide the gradient propagation. Our method dramatically outperforms current state-of-the-arts, lowering the performance gap between full-precision networks and BNNs on mainstream architectures, achieving the new state-of-the-art on binarized AlexNet (Top-1 50.5%), ResNet-18 (Top-1 59.7%), and VGG-Net (Top-1 63.2%). At inference time, Si-BNN still enjoys the high efficiency of exclusive-not-or (xnor) operations.

Introduction

Quantizing deep neural networks to the extremely low bit is an interesting yet challenging problem. There have been many good efforts to approximate full precision weights using 1-bit representation (Courbariaux, Bengio, and David 2015; Rastegari et al. 2016; Hu, Wang, and Cheng 2018). Furthermore, recent methods show that 2-bit activation and binary/ternary weight quantization can achieve promising results on ImageNet (Cai et al. 2017; Wang et al. 2018). However, the grand challenge, constraining both the weights and activations to 1 bit, has been overcome only on small datasets, such as MNIST and CIFAR-10 (Courbariaux and Bengio 2016). A better binary feature representation is of central importance in training binary neural networks on large-scale datasets.

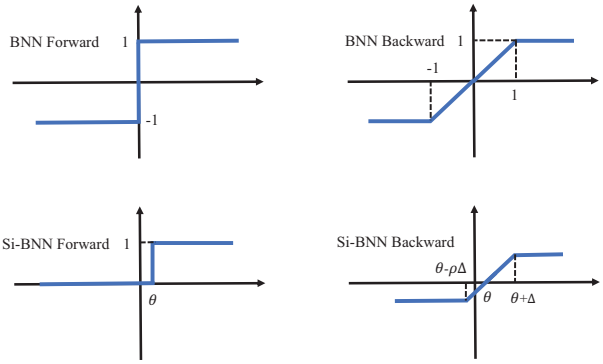


Figure 1: Forward and backward functions for BNN and Si-BNN. Our approach utilizes sparse representation instead of dense predictions in previous methods. Both θ and Δ are trainable parameters to adaptively deactivate activations and gradients. $\rho \in [0, 1]$ is the hyperparameter, which allows more gradients propagate through activated neurons (i.e., $+1$ neurons in forward propagation).

Previous BNNs (Courbariaux and Bengio 2016) mainly use $sign$ as the quantization function to turn activations into either -1 or $+1$. Moving forward along this line, XNOR-Net (Rastegari et al. 2016) further introduces a full-precision scale factor to minimize binarization error. Using this strategy, XNOR-Net first applies binarized convolutional neural networks to ImageNet classification.

Different from previous works, we start from the following question, *whether $sign$ function is optimal for network binarization?* As the name of "binary" implies, there are only two states in BNNs. From the viewpoint of least square, the simple and commonly used $sign$ function becomes the natural choice for binarization. Therefore, this question has been largely overlooked by previous works, which significantly affects the performance of BNNs.

The answer to the above question is not that straightforward. To find a better binarization function, there are several characteristics that should be satisfied: 1) The dot product of two binary vectors x and y can be implemented efficiently using bit-operations, since the ultimate goal of BNNs

*These authors contributed equally to this work.
†The corresponding author.

is to replace the computationally expensive floating-point multiply accumulations (MACs) by $xnor$ and $popcount$ operations. 2) The binarization function needs to be simple, otherwise, the profit of bit-operations can be canceled out by the bulky design. 3) A better binarization function should be more effective for network binarization than the simple $sign$ function. Only in this way, the proposal of a new binarization function is meaningful.

Inspired by the dynamic sparsity induced by ReLU, we propose the sparsity-inducing binarization, which quantizes activations to either 0 or $+1$. We reveal that the proposed method is as efficient as $sign$-based scheme with no extra computing cost, yet it achieves superior performance over dense feature representation (i.e., ± 1). By exploring the effect of sparse binary feature representation, we show that Si-BNN satisfies the above characteristics.

Since the derivative of binarization is zero almost everywhere, the appropriate gradient approximation significantly contributes to the fast training. In this work, we empower the well-known straight-through estimator (STE) with trainable thresholds to better exploit the saturation effect, which cancels the gradient when input is larger than the learned threshold. It is surprisingly simple that a linear transformation combined with sparsity-inducing binarization serves as the trainable STE.

We conduct extensive experiments on MNIST, CIFAR, and ILSVRC 2012 datasets, using the mainstream architectures such as AlexNet, ResNet, and VGG-Net. The consistent notable improvements over state-of-the-art binarization methods (Courbariaux and Bengio 2016; Rastegari et al. 2016), even multi-bit networks (Lin, Zhao, and Pan 2017; Zhu, Dong, and Su 2019) show that the proposed approach is generally effective for image categorization. We also implement a binary network execution framework, with which it is possible to run Si-BNN models over $14\times$ faster than full-precision counterparts on CPU, competitive with the popular GPU-based deep learning framework. The contributions are summarized as follows:

- Sparsity-inducing Binarized Neural Network (Si-BNN) is proposed to quantize the activations to be either 0 or $+1$, which has prominent advantages over $sign$-based binarization methods.
- Based on Si-BNN, we further introduce trainable thresholds into the backward function of binarization to guide the gradient propagation.
- Extensive experiments demonstrate that the proposed Si-BNN dramatically improves the accuracy, setting new state-of-the-arts of Binarized Neural Networks.

Related Work

Low-bit quantization (Rastegari et al. 2016; Hu et al. 2018; He and Cheng 2018) of deep neural networks has recently received increasing interest of deep learning community. It is shown that full-precision is not required in achieving high performance for deep neural networks(Gupta et al. 2015; Courbariaux, Bengio, and David 2015; Rastegari et al. 2016; Wang and Cheng 2017). By utilizing fixed-point weights representation, the model size can be dramatically reduced by an order of magnitude (up to $\sim 32\times$) without considerable loss in classification accuracy. However, some methods still use floating-point feature representation, which limits the acceleration of network computing. To further explore the advantages of fixed-point multiply accumulations, recent works focus on the optimal quantization of activations. The extremely low-bit networks with binary weights and 2-bit activations have approached floating-point level on ImageNet classification accuracy (Cai et al. 2017; Tang, Hua, and Wang 2017; Jung et al. 2019).

Beyond low-bit quantization, binary neural networks benefit from $\sim 32\times$ model compression and the replacement of expensive floating-point matrix multiplication by high computation efficiency $popcnt - xnor$ operations, which makes the runtime speed of BNN comparable with GPU. In BNN (Courbariaux and Bengio 2016), the weights and activations are firstly constrained to either $+1$ or -1, which produces reasonable results on small datasets, such as MNIST and CIFAR-10. When it comes to large-scale classification tasks, such as ImageNet (Deng et al. 2009), the accuracy drop between the full precision network and BNN is unacceptable. Although the performance of BNN can be refined by proper training strategies, such as low initial learning rate (Tang, Hua, and Wang 2017), it is still nearly 15% lower than floating-point Top-5 accuracy on AlexNet.

To improve the quality of the binary feature representation, XNOR-Net (Rastegari et al. 2016) introduces scale factors for both weights and activations during binarization process. This relaxation of binary constraint significantly contributes to the accuracy improvement on binary weights neural networks but still suffers over 10% Top-5 accuracy loss on binary activation networks. Besides that, a floating-point 2D convolution is required before binary convolution for calculating the scale factors of the activations. This step has partially reduced the advantages of efficient $popcnt - xnor$ operations.

Since weights and activations binarization leads to inference acceleration, DoReFa-Net (Zhou et al. 2016) further proposes the 2-bit gradients approximation with 1-bit weights and activations to boost training on FPGA. Due to the limited computation resources, nonlinear activation functions and its backward computing are designed to adapt the hardware platforms. Even though they achieve no significant accuracy loss on SVHN dataset with considerable energy savings, the performance on AlexNet is lower than XNOR-Net.

Multi-bit networks decompose a single convolution layer into K binary convolution operations (Lin, Zhao, and Pan 2017; Zhu, Dong, and Su 2019; Li et al. 2017; Tang, Hua, and Wang 2017). These methods achieve higher accuracy than XNOR-Net, however, at the cost of $K\times$ extra storage and computing consumption. We show that Si-BNN with $\sim 32\times$ memory saving and $\sim 64\times$ theoretical computation saving still outperforms several multi-bit networks.

As a result, while some of these methods have produced good performances on datasets such as MNIST, CIFAR-10, and SVHN, the ultimate challenges remain unsolved, such as ImageNet classification problem using binarized neural networks (i.e., one base multi-bit networks).

Approach

In this section, we first revisit the formulation of network binarization then introduce our sparsity-inducing binarization in detail including how to implement Si-BNN efficiently. Besides that, we further present trainable thresholds to better approximates the derivative of binarization function.

Network Binarization

We define a convolutional layer as a tuple $\langle \mathbf{W}, \mathbf{X} \rangle$. $\mathbf{W}^T \in \mathbb{R}^{n \times [c \cdot h \cdot w]}$, where (n, c, h, w) refers to output channels, i.e., filter number, input channels, kernel height and kernel width, respectively. $\mathbf{X} \in \mathbb{R}^{[c \cdot h \cdot w] \times [H \cdot W]}$, where (H, W) are height and width of output feature maps. In fully-connected layers, $\mathbf{W}^T$ and $\mathbf{X}$ degrade to $\mathbb{R}^{n \times c}$ and $\mathbb{R}^{c \times 1}$ respectively. The convolution or inner-product can be represented by the following equation:

$$\mathbf{Y} = \psi(\mathbf{W}^T \mathbf{X}). \tag{1}$$

where ψ represents the nonlinear activation function. This formulation introduces two inevitable problems along with the development of deep neural networks: 1) the huge computational complexity of $\mathbf{W}^T\mathbf{X}$; 2) the large storage/memory consumption of $\mathbf{W}$. Therefore, binarized neural networks which solve both problems simultaneously becomes an interesting topic in the deep learning literature.

Previous BNNs mainly use $sign$ as the quantization function, turning all weights and activations into either -1 or $+1$. Benefitting from the binary constraint, BNNs replace the floating-point multiply accumulations by the bit-level $popcnt - xnor$ operations:

$$\mathbf{w_b}^T \cdot \mathbf{x_b} = N - 2 \times popcnt(xnor(\mathbf{w_b}, \mathbf{x_b})) \tag{2}$$

where $\mathbf{w_b}$ and $\mathbf{x_b}$ represent the binary vectors of weights and activations, i.e., $\mathbf{w_b}$, $\mathbf{x_b} \in \{-1, +1\}^N$, where $N = c \cdot h \cdot w$. According to the central-limit theorem (CLT), acitvations are nearly "Gaussian" (Ioffe and Szegedy 2015; Cai et al. 2017), which is hard for $sign(\mathbf{X})$ to capture the higher-order statistics such as variance. XNOR-Net partly alleviates this problem by introducing full-precision scale factors into binarization $\psi'(x) = \alpha \cdot sign(x)$, which also facilitates the gradient straight-through estimation, i.e., $\frac{\partial \ell}{\partial \mathbf{w}_i} \approx \frac{\partial \ell}{\partial \hat{\mathbf{w}}_i}$ where $\hat{\mathbf{w}}_i = \alpha \mathbf{w_b}_i$.

Sparsity-inducing binarization

Previous activation binarization approaches commonly utilize $sign$ function to turn features into either -1 or +1, because the dot product between two -1/+1 vectors can be converted into bit-level $popcnt - xnor$ operations (Eq. 2). As shown in later sections, we show that the dot product between 0/+1 and -1/+1 vectors can still make use of $popcnt - xnor$ operations without extra computing cost. This finding enable us to further explore sparse feature representation, which have shown to be effective (Cai et al. 2017; Jung et al. 2019). Based on this insight, we propose to binarize activations to be either 0 or 1,

$$X_b = \psi(X) = \begin{cases} 1 & \text{if } X \geq \theta \\ 0 & \text{otherwise} \end{cases}, \tag{3}$$

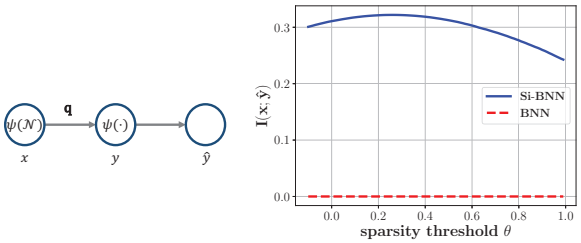


Figure 2: Mutual information between x and $\hat{y}$ in a minimal model. (Left) A three neuron binarized neural network where Gaussian inputs x after binarization, multiplied by weight $\mathbf{q} \in \{-1, +1\}$, and feeds through the same binarization $\psi(\cdot)$ yielding $\hat{y}$. (Right) Mutual information between x and $\hat{y}$ for different sparsity thresholds.

Inspired by straight-through estimator (STE), we utlize Eq. 4 for the backward gradient estimation.

$$\frac{\partial \psi}{\partial X} = \begin{cases} 1 & \text{if } 0 \leq X \leq 1 \\ 0 & \text{otherwise} \end{cases}, \tag{4}$$

The forward and backward computation is illustrated in Figure 1. It is worth noting that the θ in Eq. 3 is not necessarily 0 since the small positive values can be further quantized to 0 instead of $+1$ to minimize quantization error. In light of this, the above scheme leads to the following question: *how to determine the quantization threshold θ?*

Inspired by (Saxe et al. 2018; Tishby and Zaslavsky 2015), we focus on the influence of the neural nonlinearity on the mutual information dynamics with binary constraints, then prove that $\theta > 0$ retains more information in a minimal model. Considering a simple three neuron network shown in Figure 2 (left), the input activation sampled from a scalar Gaussian distribution $\mathcal{N}(0, 1)$ [1] passes through a binarization function $\psi(\cdot)$, which is then fed through the binarized weight $\mathbf{q}$. The hidden layer with the same binarization function yields the final output $\hat{y} = \psi(\mathbf{q} \cdot \psi(x))$.

Following the above setting, the mutual information $I(x; \hat{y})$ of two discrete random variables x and $\hat{y}$ is:

$$I(x; \hat{y}) = \sum_{x \in \mathcal{B}} \sum_{\hat{y} \in \mathcal{B}} p(x, \hat{y}) \log\left(\frac{p(x, \hat{y})}{p(x)p(\hat{y})}\right) \tag{5}$$

where $\mathcal{B}$ is either $\{-1, +1\}$ in BNNs or $\{0, +1\}$ in Si-BNN. We assume that $\mathbf{q} = sign(\mathbf{w})$ and $\mathbf{q}$ has a Rademacher distribution (based on the common weights initialization of standard Gaussian), i.e., $\frac{1}{2}(\mathbf{q}+1) \sim \text{Bernoulli}(\frac{1}{2})$. For $sign$ activation function (shown in Table 1a), it is easy to prove that $I(x; \hat{y}) = 0$ if binarized weight $\mathbf{q}$ and activation x are generally independent. This result shows that $\hat{y}$ suffers from huge information loss during the early training phase when $p(\mathbf{q}|x) \approx p(\mathbf{q})$.

As shown in Figure 2 (right), sparsity-inducing binarization $\psi(\cdot)$ keeps more information due to the existence of

[1] It has been shown that feature maps after batch-normalization are approximate normal distribution (Ioffe and Szegedy 2015; Cai et al. 2017).

Table 1: The truth table of the two-input version BNN.

(a) BNN

q \ x	−1	+1
−1	+1	−1
+1	−1	+1

(b) Si-BNN

q \ x	0	+1
−1	0	−1
+1	0	+1

zero. Following the previous assumption on $\mathbf{q}$, we obtain $p(\hat{y}=0) = p(x=0) + p(\mathbf{q}=-1)p(x=1) = p(x=0) + \frac{1}{2}(1-p(x=0))$ and $p(\hat{y}=1) = \frac{1}{2}(1-p(x=0))$ where $p(x=0)$ (p, for short) can be approximated accurately by numerical integration over $(-\infty, \theta]$. Based on Table 1b, we have the joint probability $p(x=0,\hat{y}=0) = p(\hat{y}=0|x=0)p(x=0) = p$, $p(x=0,\hat{y}=1) = 0$ and $p(x=1,\hat{y}=1) = p(x=1,\hat{y}=0) = \frac{1}{2}(1-p)$. Therefore, $I(x;\hat{y})$ can be formulated as the function of p:

$$I(x;\hat{y}) = \frac{1}{2}p\log\frac{1-p}{1+p} - \frac{1}{2}\log\frac{1-p}{1+p} + p. \tag{6}$$

Figure 2 (right) shows that the mutual information $I(x;\hat{y})$ changes along with the sparsity threshold θ. Sparsity-inducing binarization remains most informative over the range $\theta \in (0.2, 0.4)$.

Trainable binarization

Though the selection of θ has been partly solved through heuristic search, it remains unclear whether every binarization layer should share the same θ. The similar problem still exists in Eq.(4) where 0/1 clip could be suboptimal. In this section, we propose a general method to learn those hyperparameters from backward propagations.

Since the gradient estimation relies on neurons near the threshold θ, the backward function of sparsity-inducing binarization coupled with trainable thresholds is ideally

$$\frac{\partial\psi}{\partial X} = \begin{cases} 1 & \text{if } \theta - p\Delta' \le X \le \theta + q\Delta' \\ 0 & \text{otherwise} \end{cases}, \tag{7}$$

where θ, Δ' are trainable parameters. θ determines the binary threshold duiring the forward feature computation, while Δ' controls the clip interval during backward gradient estimation. The two hyperparameters p and q enable asymmetric clip interval around θ. By setting $\Delta = q\Delta'$ and $\rho = p/q$, Eq. 7 turns into

$$\frac{\partial\psi}{\partial X} = \begin{cases} 1 & \text{if } \theta - \rho\Delta \le X \le \theta + \Delta \\ 0 & \text{otherwise} \end{cases}, \tag{8}$$

where ρ becomes the only hyperparameter, and at the same time, the learnable parameters becomes θ and Δ.

To make θ and Δ trainable, we rewrite the comparisons in Eq.(8) as

$$-\rho \le \frac{1}{\Delta}(X-\theta) \le 1, \tag{9}$$

which corresponds to a simple linear transformation combined with threshold comparisons.

By setting $\hat{X} = \Delta^{-1}(X-\theta)$, the forward computation becomes

$$\psi(\hat{X}) = \begin{cases} 1 & \text{if } \hat{X} \ge 0 \\ 0 & \text{otherwise} \end{cases}, \tag{10}$$

which introdues both Δ and θ into the forward path, yet only θ serves as the threshold which contributes to the final outputs. Note that Eq.(10) performs the same function as Eq.(3) except trainable θ. In the backward propagation, our gradient estimation is simply

$$\frac{\partial\psi}{\partial\hat{X}} = \begin{cases} 1 & \text{if } -\rho \le \hat{X} \le 1 \\ 0 & \text{otherwise} \end{cases}, \tag{11}$$

which leads to the following derivatives

$$\frac{\partial\psi}{\partial X} = \frac{1}{\Delta}\cdot\begin{cases} 1 & \text{if } \theta - \rho\Delta \le X \le \theta + \Delta \\ 0 & \text{otherwise} \end{cases}, \tag{12}$$

$$\frac{\partial\ell}{\partial\Delta} = \frac{1}{\Delta^2}\sum_i \frac{\partial\ell}{\partial\hat{X}_i}(\theta - X_i), \tag{13}$$

$$\frac{\partial\ell}{\partial\theta} = -\frac{1}{\Delta}\sum_i \frac{\partial\ell}{\partial\hat{X}_i}. \tag{14}$$

It is worth noting that for (12), $\frac{\partial\psi}{\partial X}$ benefits from trainable backward interval $[\theta - \rho\Delta, \theta + \Delta]$. Besides, the term $\frac{1}{\Delta}$ further refines the gradients. When Δ becomes larger, Eq.(11) approaches the derivative of linear transformation. In this case, the gradients can be inaccurate and $\frac{1}{\Delta}$ makes a small gradient step.

Efficient computing

To apply $xnor - popcnt$ to deep neural networks, previous works quantize both activations and weights into either -1 or $+1$ (with optional scaling factors). In this section, we show that activations $1/0$ binarization is as efficient as ±1-based methods without extra computing cost. More generally, it is free to quantize activations into any two real numbers without inference speed loss.

In the l-th convolution layer of binary neural networks, we approximate the n-th filter $\mathbf{w} \in \mathbb{R}^N$ by $\beta\mathbf{w_b}$, where $N = c\cdot h\cdot w$, $\beta = \frac{1}{N}||\mathbf{w}||_1$ and $\mathbf{w_b} = sign(\mathbf{w})$. For featuremap binarization, we set activations $\mathbf{X}_l$ to be either 0 or $+1$ according to Eq.(3), and X_b can be further represented by the linear combination of ±1:

$$\begin{cases} 1 = k(+1) + b \\ 0 = k(-1) + b \end{cases} \tag{15}$$

where $k = b = \frac{1}{2}$. Similarly, for any X_b we have the unique $k, b \in \mathbb{R}$. Without loss of generality, we formulate $\psi(\mathbf{X}_l)$ as $k\mathbf{H}_l + b\mathbf{U}_l$ where $\mathbf{H}_l \in \{-1,+1\}^{[c\cdot h\cdot w]\times[H\cdot W]}$ and $\mathbf{U}_l$ denotes an all-ones matrix with the same shape as $\mathbf{H}_l$. Formally, the binary convolution is

$$\begin{aligned} \mathbf{w}^T * \mathbf{X}_l &\approx (\beta\mathbf{w_b}^T) * \psi(\mathbf{X}_l) \\ &= (\beta\mathbf{w_b}^T) * (k\mathbf{H}_l + b\mathbf{U}_l) \\ &= (\beta k)(\mathbf{w_b}^T \oplus \mathbf{H}_l) + \underbrace{(\beta b)(\mathbf{w_b}^T * \mathbf{U}_l)}_{\text{pre-computed}} \end{aligned} \tag{16}$$

where $*$ denotes floating-point convolution operation and $\oplus$ refers to a $xnor$-$popcnt$-based binary convolution. By utilizing the pre-computed $(\beta b)(\mathbf{w_b}^T * \mathbf{U}_l)$, we have the compact binary convolution function:

$$\mathbf{w}^T * \mathbf{X}_l \approx k'(\mathbf{w_b}^T \oplus \mathbf{H}_l) + b' \tag{17}$$

where k' is βk and b' refers to $(\beta b \sum_i \mathbf{w_b}_i)$. Note that both k' and b' are pre-computed before testing, i.e., Eq.17 can be implemented efficiently using $xnor$-$popcnt$ operations as $sign$-based methods

Network architecture

To achieve higher accuracy, previous network binarization methods commonly introduce a non-binary activation after binary convolution. For example, in XNOR-Net (Rastegari et al. 2016), a ReLU layer is required after each binary convolution layer, such as $Binary\text{-}Convolution + ReLU + Batch\text{-}Normalization + Binarization$. The authors of (Tang, Hua, and Wang 2017) further propose to use PReLU instead.

Since binarization also serves as a nonlinear activation function, the ideal scheme should be compatible with the original architectures. Si-BNN needs to replace the nonlinear activation (e.g., ReLU) in original networks by the proposed trainable sparsity-inducing binarization function in Eq.(10), no extra computation or storage is introduced. One of the reason is that our binarization scheme shares similar characteristics with ReLU, which significantly contributes to the great improvement on all mainstream networks without modifying the original structures.

Experiments

In this section, we evaluate the proposed Si-BNN in terms of accuracy and efficiency. Our experiments are conducted on MNIST, CIFAR-10 and ImageNet (Deng et al. 2009) datasets. Several mainstream networks such as AlexNet (Krizhevsky, Sutskever, and Hinton 2012), VGG-Net (Simonyan and Zisserman 2014), and ResNet (He et al. 2016) are used for testing.

Implementation Details

For ImageNet experiments, the input image is first proportionally resized to $256 \times N$ ($N \times 256$) with the short edge to 256. Then, subcrops of 224×224 (227×227 for AlexNet) are randomly sampled from an image or its horizontal reflection. Following XNOR (Rastegari et al. 2016) and HWGQ (Cai et al. 2017), we add a batch normalization layer after each convolution and fully-connected layer. Following (Rastegari et al. 2016), for networks with multiple fully-connected layers, such as AlexNet and VGG-Net, we place a dropout layer with $p = 0.5$ before the last layer, while no dropout is used for other networks such as ResNet. For VGG-Net, we use the same architecture as described in (Cai et al. 2017), namely VGG-Variant. For AlexNet and ResNet, we follow the setting in (Cai et al. 2017; Rastegari et al. 2016). Our binary networks are trained from scratch using Adam (Kingma and Ba 2015) with default settings. The batch size for ImageNet networks is 256. We use a weight decay of $1e^{-6}$ and momentum of 0.9 in default. Specifically, we set the weight decay of Δ and θ to zero. The initial learning rate is 0.001, then reduced by a factor of 10 at 40^{th} and 80^{th} epoch. We train all the networks on ILSVRC2012 training dataset for 100 epochs. Following the setting in previous low-bit networks (Rastegari et al. 2016), we keep the first and last layer in full precision.

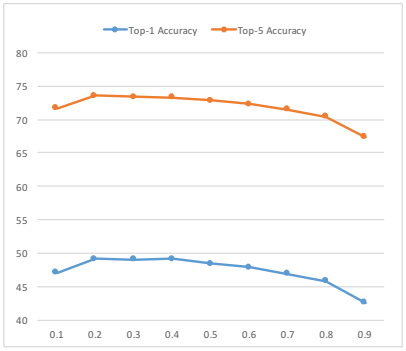


Figure 3: The accuracy on AlexNet for different sparse thresholds ranging from 0.1 to 0.9, which is consistent with the mutual information loss in Figure 2.

Table 2: Validation accuracy (%) of AlexNet on ImageNet using different ρ. "Fix" indicates the Si-BNN method with fixed hand-crafted binary threshold.

ρ	1	0.5	0.4	0.3	0.2	0.1	Fix	XNOR
Top-1	49.4	49.5	50.2	**50.5**	50.1	41.9	49.2	44.2
Top-5	73.7	73.9	74.4	**74.6**	74.6	66.8	73.4	69.2

Sparse Binarization Evaluation

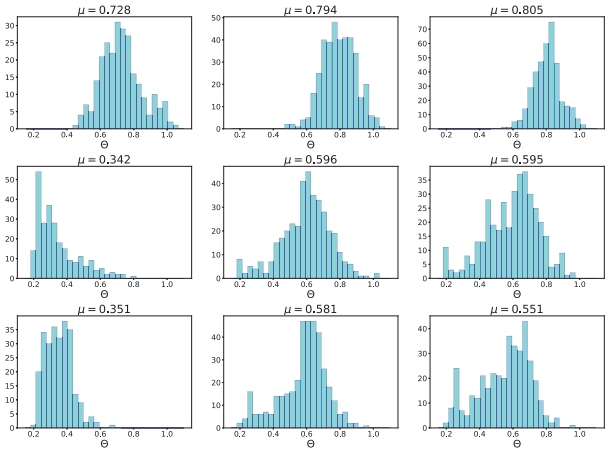
In this section, the effect of the proposed sparse binary quantization is explored, showing that sparsity has a significant effect on the performance of BNN.

The proposed Si-BNN allows us to utilize different sparsities by choosing different thresholds θ. We illustrates the impact of sparsity on the performance of Si-BNN based on AlexNet. The results are shown in Figure 3.

We can see that when the threshold is smaller than 0.4, the accuracy will increase with higher sparsity. However, when the threshold is larger than 0.4, the accuracy keeps decreasing. The experimental results of Figure 3 are consistent with our analysis in Figure 2. Another finding from Figure 3 is that when the threshold is 0.8, i.e., more than 75% activations are zeros, the accuracy can still reach 45.8%, which shows the effectiveness of the sparse property of Si-BNN. In another word, even when there are only 1/4 neurons are activated compared with previous binarization method, our Si-BNN can still get much higher accuracy than previous methods. (45.8% v.s. 44.2% of XNOR-Net).

Learnable Sparse Binarization

In the previous section, we have evaluated the performance under different sparsity selected by hand, which has demonstrated that the sparse binarization has significant performance improvement over non-sparse binarization. However,



(a) After Conv2 (b) After Conv3 (c) After Conv4

Figure 4: The histogram of θ for trainable binarization functions after convolution layers in AlexNet. Three rows, from top to bottom, correspond to $\rho = 0.1, 0.3, 1.0$ respectively. μ is the mean value of θ across different channels (each channel has its own θ). Shallow layers require dense representation while deeper layers tend to have higher sparsity.

the hand-selected sparse threshold may not be optimal. In this section, we evaluate the proposed trainable sparse binarization method under different hyperparameters ρ.

In light of the empirical success of $\theta \in [0.2, 0.4]$, we clip θ which is less than 0.2. As shown in Table 2, trainable thresholds consistently outperform hand-crafted threshold. Especially for $\rho = 0.3$, the top-1 accuracy improves by 1.3%, showing that adaptive thresholds have significant advantages over the fixed threshold. Overall, the Si-BNN with trainable sparse binarization outperforms previous leading method XNOR-Net by 6.3% top-1 accuracy.

To further understand the effect of the learnable binarization, we present the distribution of the learned θ after training under different ρ, as shown in Figure 4. Note that though ρ only controls the clip interval during backward propagation of Si-BNN, it still affects the distribution of θ during training. From Figure 4, it is noticed that the distribution of θ changes along with layer going deeper When ρ becomes smaller (the from bottom to top row), Si-BNN tends to generate more sparse feature maps, i.e., larger μ and θ. Generally, low-level features (e.g., Conv2) require dense representation, yet high-level features (e.g., Conv4) enjoy the sparsity where neurons corresponding to indiscriminative regions are deactivated.

Network Binarization Results

In this section, we evaluate the Si-BNN with learnable sparse thresholds, by comparing with the state-of-the-art low-bit networks on various architectures. It is shown that our method notably outperforms previous BNN methods, even comparable to multi-bit networks and 2-bit activation schemes. In appendix, we also visualize the feature map of Si-BNN, it is more interpretable than $sign$-based method with clearer saliency maps.

Table 3: The error rates on MNIST. Bit-width for activations and weights (before and after "+") are reported respectively.

Bit-width	Method	Error(%)
1+1	BNN (Courbariaux and Bengio 2016)	1.40
1+1	LAB2 (Hou, Yao, and Kwok 2017)	1.38
1+1	**Si-BNN**	**1.26**
1+32	LAB (Hou, Yao, and Kwok 2017)	1.18
1+32	BC (Courbariaux, Bengio, and David 2015)	1.29
32+32	Float (Courbariaux and Bengio 2016)	1.19

Table 4: The error rates on CIFAR-10. Bit-width for activations and weights (before and after "+") are reported respectively.

Bit-width	Method	Error(%)
1+1	BNN (Courbariaux and Bengio 2016)	10.15
1+1	XNOR (Rastegari et al. 2016)	10.17
1+1	LAB2 (Hou, Yao, and Kwok 2017)	12.28
1+1	**Si-BNN**	**9.80**
1+32	BC (Courbariaux, Bengio, and David 2015)	9.86
1+32	BWN (Rastegari et al. 2016)	9.88
1+32	LAB (Rastegari et al. 2016)	10.50
32+32	Float (Courbariaux and Bengio 2016)	10.94

Results on MNIST We first conduct experiments on MNIST dataset. For fair comparisons, we use the same network architecture described in BNN, which consists of 3 hidden layers of 2048 binary units (i.e., binarized MLP) with softmax loss. The Si-BNN results compared with the state-of-the-art are listed in Table 3.

Results on CIFAR-10 In addition, we evaluate Si-BNN on CIFAR-10 dataset. For fair comparisons, we set the network architecture to be identical as BNN and XNOR. No extra preprocessing was used except the standard horizontal flip. At both training and testing time, we only use the original 32×32 color images. Table 4 shows the Si-BNN result compared with the state-of-the-art. This is consistent with ImageNet results, and our approach outperforms all previous binarization methods like BNN and XNOR by a large margin.

Results on ImageNet To further illustrate the effectiveness of the proposed approach, we compare Si-BNN with current state-of-the-art methods on ImageNet dataset. The comparison results based on AlexNet and ResNet-18 are shown in Table 5 and Table 6, respectively. These results show that the Si-BNN outperforms the best previous binary methods by 2.6% and 5.3% on AlexNet and ResNet-18.

To fully illustrate the superiority of Si-BNN over previous works, we also compare Si-BNN with strong baseline methods such as multi-bit networks (i.e., replace a full precision convolution layer by several binarized convolution layers) and 2-bit networks. As listed in Table 5, the performance of Si-BNN on AlexNet notably outperforms recent leading methods and it is comparable to even 2-bit weights and 2-bit activation methods. The improvements of ResNets, listed in Table 6, are consistent with AlexNet. Note that Bi-Real (Liu et al. 2018) proposes to keep 1×1 downsampling layers

Table 5: Comparison with the state-of-the-art methods on AlexNet. "-" indicates the accuracy is not reported. "×" means multi-bit networks with multi-branch.

Method	Weight	Activation	Top-1	Top-5
BNN (Courbariaux and Bengio 2016)	1	1	27.9	50.4
DOREFA (Zhou et al. 2016)	1	1	43.6	–
XNOR (Rastegari et al. 2016)	1	1	44.2	69.2
XNOR + Distribution Loss (Ding et al. 2019)	1	1	47.8	71.5
Quantization Network (Yang et al. 2019)	1	1	47.9	72.5
Si-BNN	1	1	**50.5**	**74.6**
CompactNet (Tang, Hua, and Wang 2017)	1	2	46.6	71.1
WRPN (Mishra et al. 2018)	1×2	1×2	48.3	–
DOREFA (Zhou et al. 2016)	1	2	49.8	–
BENN-SB-3, Bagging (Zhu, Dong, and Su 2019)	1×3	1×3	48.8	–
BENN-SB-3, Boosting (Zhu, Dong, and Su 2019)	1×3	1×3	50.2	–
WEQ (Park, Ahn, and Yoo 2017)	2	2	50.6	75.0

Table 6: Comparison with the state-of-the-art methods using ResNet-18 on ImageNet. "-" indicates the accuracy is not reported. "*" means ResNet with Bi-Real-like double skip connections, without full precision 1×1 downsampling convolutions. "×" refers to multi-bit networks with multi-branch or multi-network.

Network	Method	Weights	Activation	Top-1	Top-5
ResNet-18	BNN (Courbariaux and Bengio 2016)	1	1	42.2	67.1
	ABC (Lin, Zhao, and Pan 2017)	1	1	42.7	67.6
	XNOR (Rastegari et al. 2016)	1	1	51.2	73.2
	QNet (Yang et al. 2019)	1	1	53.6	75.3
	BENN-6, Bagging (Zhu, Dong, and Su 2019)	1×6	1×6	57.9	–
	Si-BNN	1	1	**58.9**	**81.3**
ResNet-18*	Bi-Real (Liu et al. 2018)	1	1	56.4	79.5
	Si-BNN	1	1	**59.7**	**81.8**

Table 7: The accuracy of Si-BNN (binary weights and **binary** activations) on various ImageNet networks, compared with binary weights and **2-bit** activation method (HWGQ) and full precision baselines.

Model		Baselines	Si-BNN	HWGQ
AlexNet	Top-1	60.4	50.5	52.7
	Top-5	82.5	74.6	76.3
VGG-Variant	Top-1	69.8	63.2	64.1
	Top-5	89.7	85.0	85.7
ResNet-18	Top-1	69.3	59.7	59.6
	Top-5	89.2	81.8	82.2
ResNet-34	Top-1	73.3	63.3	64.3
	Top-5	91.4	84.4	85.7

in full-precision for preserving the performance. To avoid the extra computing cost and floating-point parameters, we replace the convolution operation with max pooling. Even with fewer parameters and computation, Si-BNN surpasses the current best-performing ensemble method (Zhu, Dong, and Su 2019).

From Table 7, *it is the first time that a binarized neural network can be competitive with the 2-bit activation methods (HWGQ (Cai et al. 2017)) on several mainstream network architectures*. Besides that, our VGG-Variant model is approaching full precision baseline with only 4.7% Top-5 accuracy loss, which could be a huge step towards real-world applications. Overall, our method is generally effective for the most challenging image categorization task.

Conclusion

In this paper, we propose a simple yet effective network binarization method, Sparsity-Inducing Binarized Neural Networks (Si-BNN). Through simple affine transformations, we show that quantizing activations to 0 and $+1$ can enjoy the sparse feature representation and still benefit from the high-efficiency $xnor$-$popcnt$ operations with no extra computing cost. Besides that, we introduce trainable thresholds into the forward and backward propagation of sparsity-inducing binarization, which contributes to the state-of-the-arts on most mainstream architectures using BNN. Experiments on ImageNet, MNIST and FICAR-10 benchmarks demonstrate that our Si-BNN dramatically outperforms current best-performing methods, lowering the performance gap between full-precision networks and binarized neural networks.

Acknowledgments

This work was supported in part by National Natural Science Foundation of China (No.61972396, 61876182, 61906193), the Strategic Priority Research Program of Chinese Academy of Science(No.XDB32050200), the Advance Research Program (31511130301).

References

Cai, Z.; He, X.; Sun, J.; and Vasconcelos, N. 2017. Deep learning with low precision by half-wave gaussian quantization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Courbariaux, M., and Bengio, Y. 2016. Binarized neural networks. In *Advances in neural information processing systems*, 4107–4115.

Courbariaux, M.; Bengio, Y.; and David, J. 2015. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in neural information processing systems*, 3123–3131.

Deng, J.; Dong, W.; Socher, R.; Li, L.; Li, K.; and Li, F. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*, 248–255.

Ding, R.; Chin, T.-W.; Liu, Z.; and Marculescu, D. 2019. Regularizing activation distribution for training binarized deep networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Gupta, S.; Agrawal, A.; Gopalakrishnan, K.; and Narayanan, P. 2015. Deep learning with limited numerical precision. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, 1737–1746.

He, X., and Cheng, J. 2018. Learning compression from limited unlabeled data. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 752–769.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, 770–778.

Hou, L.; Yao, Q.; and Kwok, J. T. 2017. Loss-aware binarization of deep networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Hu, Q.; Li, G.; Wang, P.; Zhang, Y.; and Cheng, J. 2018. Training binary weight networks via semi-binary decomposition. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 637–653.

Hu, Q.; Wang, P.; and Cheng, J. 2018. From hashing to cnns: Training binary weight networks via hashing. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018*.

Ioffe, S., and Szegedy, C. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, 448–456.

Jung, S.; Son, C.; Lee, S.; Son, J.; Han, J.-J.; Kwak, Y.; Hwang, S. J.; and Choi, C. 2019. Learning to quantize deep networks by optimizing quantization intervals with task loss. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Kingma, D. P., and Ba, J. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 1097–1105.

Li, Z.; Ni, B.; Zhang, W.; Yang, X.; and Gao, W. 2017. Performance guaranteed network acceleration via high-order residual quantization. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, 2603–2611.

Lin, X.; Zhao, C.; and Pan, W. 2017. Towards accurate binary convolutional neural network. In *Advances in neural information processing systems*, 344–352.

Liu, Z.; Wu, B.; Luo, W.; Yang, X.; Liu, W.; and Cheng, K. 2018. Bi-real net: Enhancing the performance of 1-bit cnns with improved representational capability and advanced training algorithm. In *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XV*, 747–763.

Mishra, A. K.; Nurvitadhi, E.; Cook, J. J.; and Marr, D. 2018. WRPN: wide reduced-precision networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

Park, E.; Ahn, J.; and Yoo, S. 2017. Weighted-entropy-based quantization for deep neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, 7197–7205.

Rastegari, M.; Ordonez, V.; Redmon, J.; and Farhadi, A. 2016. Xnor-net: Imagenet classification using binary convolutional neural networks. In *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part IV*, 525–542.

Saxe, A. M.; Bansal, Y.; Dapello, J.; Advani, M.; Kolchinsky, A.; Tracey, B. D.; and Cox, D. D. 2018. On the information bottleneck theory of deep learning. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

Simonyan, K., and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *CoRR* abs/1409.1556.

Tang, W.; Hua, G.; and Wang, L. 2017. How to train a compact binary neural network with high accuracy? In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, 2625–2631.

Tishby, N., and Zaslavsky, N. 2015. Deep learning and the information bottleneck principle. In *2015 IEEE Information Theory Workshop, ITW 2015, Jerusalem, Israel, April 26 - May 1, 2015*, 1–5.

Wang, P., and Cheng, J. 2017. Fixed-point factorized networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, 3966–3974.

Wang, P.; Hu, Q.; Zhang, Y.; Zhang, C.; Liu, Y.; and Cheng, J. 2018. Two-step quantization for low-bit neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Yang, J.; Shen, X.; Xing, J.; Tian, X.; Li, H.; Deng, B.; Huang, J.; and Hua, X.-s. 2019. Quantization networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Zhou, S.; Ni, Z.; Zhou, X.; Wen, H.; Wu, Y.; and Zou, Y. 2016. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv* abs/1606.06160.

Zhu, S.; Dong, X.; and Su, H. 2019. Binary ensemble neural network: More bits per network or more networks per bit? In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.